Open Journal of Modelling and Simulation, 2016, 4, 1-15
Published Online January 2016 in SciRes. http://www.scirp.org/journal/ojmsi
http://dx.doi.org/10.4236/ojmsi.2016.41001

On the Convergence of the Dual-Pivot Quicksort Process

Mahmoud Ragab[1], Beih El-Sayed El-Desouky[2], Nora Nader[2]

[1]Department of Mathematies, Faculty of Science, Al Azhar University, Cairo, Egypt
[2]Department of Mathematics, Faculty of Science, Mansoura University, Mansoura, Egypt
Email: meragab23@yahoo.mail, bdesouky@mans.edu.eg, scinora2012@yahoo.com

Received 19 November 2015; accepted 26 January 2016; published 29 January 2016

Abstract

Sorting an array of objects such as integers, bytes, floats, etc. is considered as one of the most important problems in Computer Science. Quicksort is an effective and wide studied sorting algorithm to sort an array of *n* distinct elements using a single pivot. Recently, a modified version of the classical Quicksort was chosen as standard sorting algorithm for Oracles Java 7 routine library due to Vladimir Yaroslavskiy. The purpose of this paper is to present the different behavior of the classical Quicksort and the Dual-pivot Quicksort in complexity. In Particular, we discuss the convergence of the Dual-pivot Quicksort process by using the contraction method. Moreover we show the distribution of the number of comparison done by the duality process converges to a unique fixed point.

Keywords

Randomized Quicksort, Convergence, Dual-Pivot Quicksort Process, Running Time Analysis

1. Introduction

Quicksort is one of the important sorting algorithms. Hoare [1] proposed an algorithm depended on selecting an arbitrary element from the array. This element called a pivot element such that Quicksort algorithm used for parting the arrays into two sub-arrays: those smaller than the pivot and those larger than the pivot [2].

After that Quicksort depends on recursive sorting of the two subarrays. Later Sedgewick studied several variants.

Regnier [3] studied the limiting distribution of the number of comparisons done by Quicksort algorithm when suitably normalized. It converges with uncertain unknown limit. The first accounts were computed by Hennequin who proved that this distribution is not a normal distribution. The limiting distribution is characterized by a stochastic fixed point equation [4] [5]. The cost of the Quicksort algorithm depends on the position of the selected pivot. There are many cases to choose the pivot element. The worst-case, the best-case and the average case express the performance of the algorithm. We will discuss some of them and for more details; we refer to Ragab [6] and [7]. The worst-case occurs when the pivot is the smallest (or largest) element at partitioning on array of size n, yielding one empty sub-array, one element (pivot) in the correct place and one sub-array of size $n - 1$. So, the two sub-arrays are lopsided so this case is defined by worst case [8]. We found the recursion depth is $n - 1$ levels and the complexity of Quicksort is $O\left(n^2\right)$. The best case occurs when the pivot is in the median at each partition step, *i.e.* after each partitioning, on array of size n, yielding two sub-arrays of approximately equal size and, the pivot element in the middle position takes n data comparisons [9]. There are various methods to choose a good pivot, like choosing the First element, Last element, and Median-of-three elements (selection three elements, and find the median of these elements), and so on. In this case the Quicksort algorithm selects a pivot by random selection each time. This choice reduces probability that the worst-case ever occurs. The other method, which essential prevents the worst case from ever occurring, picks a pivot as the median of the array each time. When we chose the pivot, we compare all other elements to it and we have $n - 1$ comparisons to divided the array. The choosing of the pivot divided the array into one sub-array of size 0 and one sub-array of size $n - 1$, or into a sub-array of size 1 and other one of size $n - 2$, and so on up to a sub-array of size $n - 1$ and one of size 0. We have n possible positions and each one is equality in probability $1/n$. Hennequin studied comparisons for array by using Quicksort with r pivots when $r = 2$, same comparisons as classic Quicksort in one partitioning. When $r > 2$, he found the problem is complied. Yaroslavskiy [10] introduced a new implementation of Dual-pivot Quicksort in Java 7's runtime library. In 2012, Wild and Nabel denoted exact numbers of swaps and comparisons for Yaroslavskiy's algorithm [10]. In this paper, our aim is to analyze the running time performance of Dual-pivot Quicksort. The limiting distribution of the normalized number of comparisons required by the Dual-pivot Quicksort algorithm is studied. It is known to be the unique fixed point of a certain distributional transformation T with zero mean and finite variance.

We show that using two pivot elements (or partitioning to three subarrays) is very efficient, particularly on large arrays. We propose the new Dual-pivot Quicksort scheme, faster than the known implementations, which improves this situation (see in [11] and [12]). The implementation of the Dual-pivot Quicksort algorithm has been inspected on different inputs and primitive data types.

The new Quicksort algorithm uses partitioning a source array $T\left[\,\right]g$, where g is primitive array which we need to sort it. Such as int, float, byte, char, double, long and short, to three parts defined by two pivot elements p and q (and therefore, there are pointers A, B, C and left and right indices of the first and last elements respectively). The aim of this paper is topresent such a version arising from an algorithm depending on the work in [13] and [14]. The Dual-pivot Quicksort is explained clearly in [15] and it works as follow:

1) For small arrays (length < 17), use the Insertion sort algorithm [10].

2) Choose two pivot elements p and q. We can get, for example, the first element $g\left[left\right]$ as p and the last element $g\left[right\right]$ as q.

3) p must be less than q, otherwise they are swapped. So, we have the following parts.

- **Part I** with indices from *left* + 1 to $A - 1$ with elements, which are less than p.
- **Part II** with indices from A to $B - 1$ with elements, which are greater or equal to p and less or equal to q.
- **Part III** with indices from $C + 1$ to *right* − 1 with elements greater than q.
- **Part IV** contains the rest of the elements to be examined with indices from B to C.

4) The next element $g\left[B\right]$ from the part IV is compared with two pivots p and q, and placed to the corresponding part I, II, or III.

5) The pointers A, B, and C are changed in the corresponding directions.

6) The steps 4 - 5 are repeated while $B \le C$.

7) The pivot element p is swapped with the last element from part I, the pivot element q is swapped with the first element from part III.

8) The steps 1 - 7 are repeated recursively for every part I, part II, and part III as in **Figure 1**.

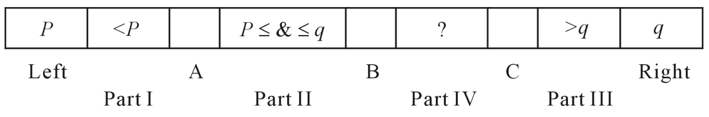

Figure 1. Graph explains the dual-pivot quicksort algorithm.

2. Run-Time Performance

In this section, we introduce some running time of the Dual-pivot Quicksort. An efficient procedure is described by Vasileios Iliopoulos and David B. Penman [13], where they analyzethe Dual pivot Quicksort algorithm. Their approach can be here provided and for more detailswe refer to [13] and [14]. First we introduce the algorithm of it and we compare between it and the classical Quicksort as follows [16].

The following graphs show the relation between the size of array which need to sort and the time of complexity which represent by the number of comparisons and swaps as in **Figure 2**. We found the Dual-pivot Quicksort is faster than classical Quicksort.

3. The Dual-Pivot Quicksort Average Case Analysis

To find the distributional equation, we note the following: for the underlying process, there are two parts. The first part is partitioning and the second is the total number of comparisons to sort an array of $n \geq 2$ keys, when the pivot is a uniform random variable $\{1,2,3,\cdots,n\}$ is equal to the number of comparisons to sort the sub-array of on $U_{n_1}-1$ keys below the first pivot [17].

In addition, we need to compute the number of comparisons to sort the sub-array of $n-U_{n_2}$ elements above the second pivot plus the number of comparisons to sort the sub-array of $U_{n_2}-U_{n_1}-1$ elements between the first and the second pivot.

Plus $2n-U_{n_1}-2$ comparisons done to partition the array which come from when the all elements compare one time with the first pivot and the remain elements compare two times with the second and the first pivot. Therefore,

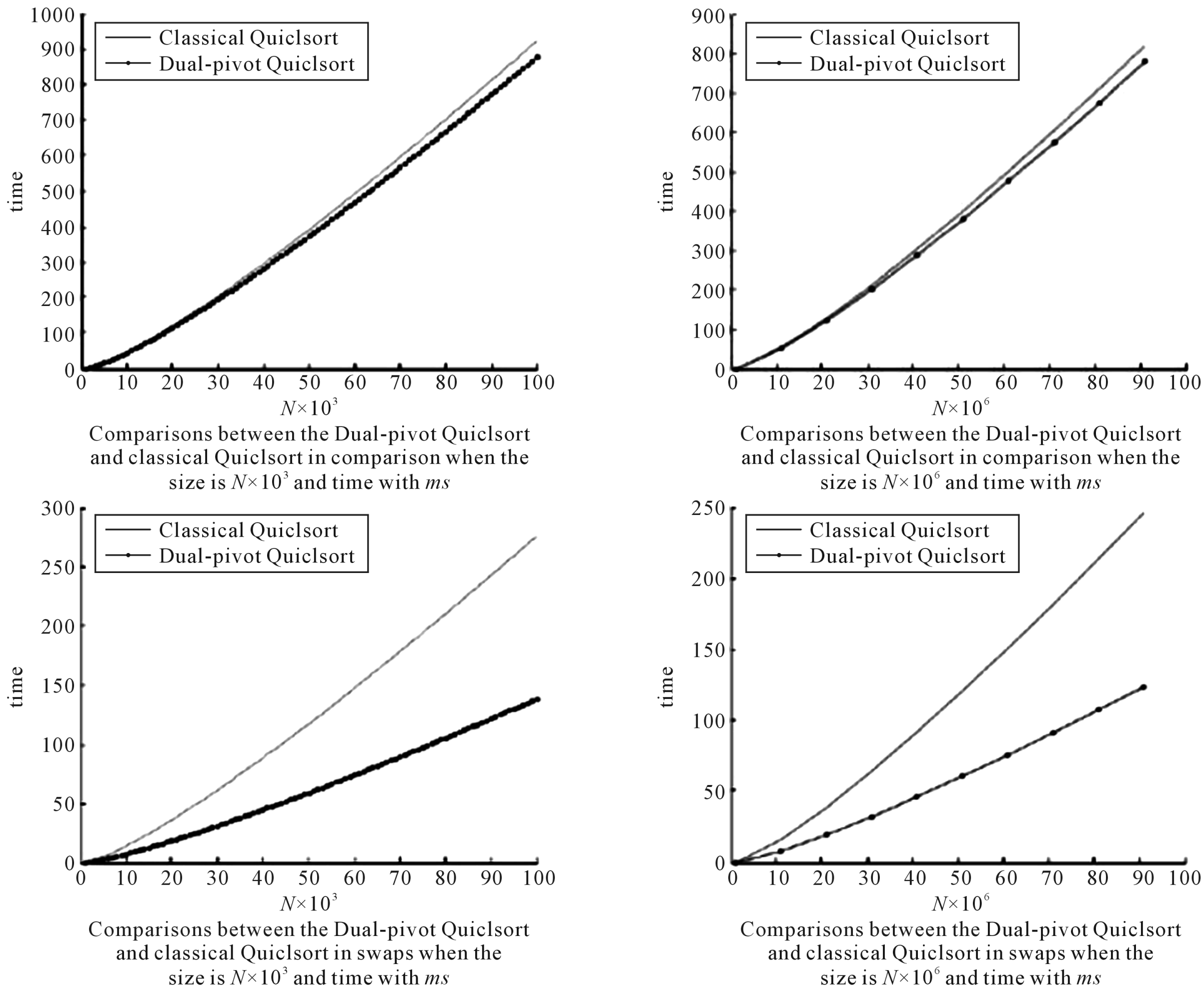

Figure 2. Comparison between the classical Quicksort and the Dual-pivot Quicksort.

$$X_n \overset{d}{=} 2n - U_{n_1} - 2 + X_{U_{n_1}-1} + X^*_{n-U_{n_2}} + \bar{X}_{U_{n_2}-U_{n_1}-1}, \tag{1}$$

where the random variables $X_{U_{n_1}-1}$, $X^*_{n-U_{n_2}}$ and $\bar{X}_{U_{n_2}-U_{n_1}-1}$ are identically distributed and independent of U_{n_1} and U_{n_2}. Here $\overset{d}{=}$ refers to the equality in distribution.

The array is partitioned into three subarrays one with $U_{n_1}-1$ keys smaller than the first pivot, a subarray of $U_{n_2}-U_{n_1}-1$ keys between two pivots and the part of $n-U_{n_2}$ elements greater than the second pivot. The algorithm is then recursively applied to each of these subarrays. The number of comparisons during the first stage is

$$A_n = 1 + \left[\left(U_{n_1}-1\right) + 2\left(U_{n_2}-U_{n_1}-1\right) + 2\left(n-U_{n_2}\right)\right],$$

where $U_{n_1} = 1,\cdots,n-1$ and $U_{n_2} = U_{n_1}+1,\cdots,n$. Using [11], the average value of A_n can be calculated as follow:

$$\begin{aligned} E\left(A_n\right) &= \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(1 + \left[\left(i-1\right) + 2\left(j-i-1\right) + 2\left(n-j\right)\right]\right) \\ &= \frac{1}{\binom{n}{2}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(2n-i-2\right) = \frac{2}{n\left(n-1\right)} \left(\frac{5}{6}n^3 - 2n^2 + \frac{7}{6}n\right) = \frac{5n-7}{3} \end{aligned}$$

4. Expected Number of Comparisons

Here by Equation (1) and using [13], it is easy to determine the recurrence for the expected number of comparisons due to the duality as follow:

$$E\left(X_n\right) = \frac{5n-7}{3} + \frac{2}{n\left(n-1\right)} \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E\left(X_i - 1\right) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E\left(X^*_{n-U_{n_2}}\right) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E\left(\bar{X}_{j-i-1}\right) \right).$$

Since the three double sums above are equal, then the recurrence becomes

$$E\left(X_n\right) = \frac{5n-7}{3} + \frac{6}{n\left(n-1\right)} \sum_{i=1}^{n-1} \left(n-i\right) E\left(X_{i-1}\right),$$

setting $a_n = E\left(X_n\right)$,

$$a_n = \frac{5n-7}{3} + \frac{6}{n\left(n-1\right)} \sum_{i=1}^{n-1} \left(n-i\right) a_{i-1},\ n \ge 2.$$

By initial conditions we have $a_0 = a_1 = 0$. Multiplying both sides by $\binom{n}{2}$, we obtain

$$\binom{n}{2} a_n = \binom{n}{2} \left(\frac{5n-7}{3} + \frac{6}{n\left(n-1\right)} \sum_{i=1}^{n-1} \left(n-i\right) a_{i-1} \right) = \frac{n\left(n-1\right)\left(5n-7\right)}{6} + 3\sum_{i=1}^{n-1} \left(n-i\right) a_{i-1}.$$

We introduce a difference operator for the solution of this recurrence. The operator is defined by

$$\Delta F\left(n\right) := \Delta F\left(n+1\right) - F\left(n\right). \tag{2}$$

And for higher orders

$$\Delta^k F\left(n\right) := \Delta^{k-1} F\left(n+1\right) - \Delta^{k-1} F\left(n\right).$$

Thus, we have

$$\Delta\binom{n}{2}a_n = \binom{n+1}{2}a_{n+1} - \binom{n}{2}a_n = \frac{5n^2-3n}{2} + 3\sum_{i=0}^{n-1}a_i.$$

$$\Delta\binom{n}{2}a_n = \Delta\binom{n+1}{2}a_{n+1} - \Delta\binom{n}{2}a_n = 5n+1+3a_n.$$

By definition (2),

$$\Delta^2\binom{n}{2}a_n = \Delta\binom{n+1}{2}a_{n+1} - \Delta\binom{n}{2}a_n = \binom{n+2}{2}a_{n+2} - 2\binom{n+1}{2}a_{n+1} + \binom{n}{2}a_n.$$

$$(n+1)(n+2)a_{n+2} - 2n(n+1)a_{n+1} + n(n-1)a_n = 2(5n+1+3a_n)$$

then

$$(n+1)\big((n+2)a_{n+2} - (n-2)a_{n+1}\big) + (n+2)\big((n+1)a_{n+1} - (n-3)a_n\big) = 2(5n+1).$$

Dividing by $(n+1)(n+2)$, we obtain the telescoping recurrence

$$\frac{(n+2)a_{n+2} - (n-2)a_{n+1}}{n+2} = \frac{(n+1)a_{n+1} - (n-3)a_n}{n+1} + \frac{2(5n+1)}{(n+1)(n+2)},$$

which yields

$$\frac{(n+2)a_{n+2} - (n-2)a_{n+1}}{n+2} = 2\sum_{j=0}^{n}\frac{5j+1}{(j+1)(j+2)} = \frac{18}{n+2} + 10H_{n+1} - 18n.$$

Multiplying by $\frac{(n-1)(n-2)(n-3)}{24}$, this recurrence is transformed to a telescoping one

$$\binom{n}{4}a_n = \binom{n-1}{4}a_{n-1} + \frac{18(n-1)(n-2)(n-3)}{24} + 10\binom{n}{4}H_{n-1} - 18\binom{n}{4}$$

$$\binom{n}{4}a_n = 18\sum_{j=1}^{n}\frac{(j-1)(j-2)(j-3)}{24} + 10\sum_{j=1}^{n}\binom{j}{4}H_{j-1} - 18\sum_{j=1}^{n}\binom{j}{4}. \qquad (3)$$

By using maple V. Iliopoulos and D. B. Penman [13] get

$$\sum_{j=1}^{n}(j-1)(j-2)(j-3) = 6\binom{n}{4}.$$

And for the other sums in Equation (3):

$$\sum_{j=1}^{n}\binom{j}{4}H_j = \binom{n+1}{5}\left(H_{n+1} - \frac{1}{5}\right).$$

Therefore,

$$\begin{aligned}\sum_{j=1}^{n}\binom{j}{4}H_{j-1} &= \sum_{j=1}^{n}\binom{j}{4}\left(H_j - \frac{1}{j}\right) = \sum_{j=1}^{n}\binom{j}{4}H_j - \sum_{j=1}^{n}\binom{j}{4}\frac{1}{j}\\ &= \binom{n+1}{5}\left(H_{n+1} - \frac{1}{5}\right) - \frac{1}{24}\sum_{j=1}^{n}(j-1)(j-2)(j-3)\\ &= \binom{n+1}{5}\left(H_{n+1} - \frac{1}{5}\right) - \binom{n}{4}\frac{1}{4}.\end{aligned}$$

Now the equation becomes

$$\binom{n}{4}a_n = \frac{9}{2}\binom{n}{4} + 10\left[\binom{n+1}{5}\left(H_{n+1} - \frac{1}{5}\right) - \binom{n}{4}\frac{1}{4}\right] - 18\binom{n+1}{5}$$

$$a_n = \frac{9}{2} + 10\left[\frac{n+1}{5}\left(H_{n+1} - \frac{1}{5}\right) - \frac{1}{4}\right] - 18\frac{n+1}{5}.$$

Finally, the expected number of comparisons, when two pivots are chosen is

$$a_n = 2(n+1)H_n - 4n \sim 2n\log(n), \tag{4}$$

where H_n is the harmonic number defined by $H_n := \sum_{k=1}^{n}\frac{1}{k}$ (see [18] and [19]).

This is the same value of the expected number of comparisons, when one pivot chosen in the classical Quicksort [20]. Note that this result for the dual Quicksort is identical with theexpected number of comparisons in [13].

5. Varience of Comparisons

The main result of this section was obtained by [13] (see following results for explanationand notation). Now we compute the variance of the number of comparisons by Dual-pivot Quicksort. Recall that

$$A_n = 2n - i - 2 \text{ and } E(A_n) = \frac{5n-7}{3}. \tag{5}$$

From Equation (1), we have

$$P(X_n = t) = \frac{1}{\binom{n}{2}}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P\left(A_n + X_{i-1} + \bar{X}_{j-i-1} + X^*_{n-j} = t\right),$$

noting that the resulting subarrays are independently sorted, then we get

$$P(X_n = t) = \frac{1}{\binom{n}{2}}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{l,m}\left[P(X_{i-1} = l)P\left(\bar{X}_{j-i-1} = m\right)P\left(X^*_{n-j} = t - m - l - 2n + i + 2\right)\right].$$

Letting

$$f_n(z) = \sum_{t=0}^{\infty} P(X_n = t)z^t,$$

be the ordinary probability generating function for the number of comparisons needed to sort *n* keys, we obtain

$$f_n(z) = \frac{1}{\binom{n}{2}}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} z^{2n-i-2} f_{i-1}(z) f_{j-i-1}(z) f_{n-j}(z) \tag{6}$$

It holds that $f_n(1) = 1$ and $f''_n(1) = 2(n+1)H_n - 4n$. Moreover, the second order derivative of Equation (6) evaluated at $z = 1$ is recursively given by

$$\begin{aligned} f''_n(z) = \frac{2}{n(n-1)}\Bigg[&\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(2n-i-2)^2 - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(2n-i-2) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(2n-i-2)E(X_{i-1}) \\ &+ 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(2n-i-2)E\left(X_{j-i-1}\right) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(2n-i-2)E\left(X_{n-j}\right) \\ &+ 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}E(X_{i-1})E\left(X_{j-i-1}\right) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}E(X_{i-1})E\left(X_{n-j}\right) + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}E\left(X_{j-i-1}\right)E\left(X_{n-j}\right) \\ &+ \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} f''_{i-1}(1) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} f''_{i-j-1}(1) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} f''_{n-j}(1)\Bigg]. \end{aligned}$$

By simple manipulation of indices, the sums of the products of expected values are equal. The double sum of the product of the mean number of comparisons can be simplified as follows:

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}E\left(X_{i-1}\right)E\left(X_{n-j}\right)=\sum_{i=1}^{n-1}E\left(X_{i-1}\right)\sum_{j=0}^{n-i-1}E\left(X_{j}\right)$$
$$=\sum_{i=1}^{n-1}\left[2(i+1)H_{i-1}-4(i-1)\right]\left[2\binom{n-i+1}{2}+\frac{n-i-5(n-i)^2}{2}\right].$$

We find

$$\sum_{i=1}^{n-1}i\binom{n-i+1}{2}H_{i-1}H_{n-i}=\sum_{i=1}^{n-1}\left((i-1)+1\right)\binom{n-i+1}{2}H_{i-1}H_{n-i}$$
$$=\sum_{i=1}^{n-1}(i-1)\binom{n-i}{2}H_{i-1}H_{n-i}+\sum_{i=1}^{n-1}\binom{n-i}{2}H_{i-1}H_{n-i}$$
$$+\sum_{i=1}^{n-1}(i-1)(n-i)H_{i-1}H_{n-i}+\sum_{i=1}^{n-1}(n-i)H_{i-1}H_{n-i}.$$

The recurrence becomes

$$f_n''(1)=2(n+1)(n+2)\left(H_n^2-H_n^{(2)}\right)-H_n\left(\frac{17}{3}n^2+\frac{47}{3}n+6\right)$$
$$+\frac{209}{36}n^2+\frac{731}{36}n+\frac{13}{6}+\frac{6}{n(n-1)}\sum_{i=1}^{n-1}(n-i)\,f_{i-1}''(1),$$

where $H_n^{(2)}$ is the second order harmonic number defined by $H_n^{(2)}:=\sum_{k=1}^{n}\frac{1}{k^2}$ (see [17] and [18]). Letting $d_n=f_n''(1)$ and subtracting $\binom{n}{2}d_n$ from $\binom{n+1}{2}d_{n+1}$, we get

$$\Delta\binom{n}{2}d_n=4n(n+1)(n+2)\left(H_n^2-H_n^{(2)}\right)-\frac{nH_n}{9}\left(84n^2+198n+42\right)$$
$$+3\sum_{i=1}^{n}d_{i-1}+\frac{n}{9}\left(79n^2+231n+14\right).$$

By using the identity [4]

$$H_{n+1}^2-H_{n+1}^{(2)}=\left(H_n^2-H_n^{(2)}+\frac{2H_n}{n+1}\right). \tag{7}$$

It holds that

$$\Delta^2\binom{n}{2}d_n=12(n+1)(n+2)\left(H_n^2-H_n^{(2)}\right)-H_n\left(20n^2+32n-12\right)+17n^2+37n+3d_n.$$

The previous equation is the same as

$$\binom{n+2}{2}d_{n+2}-2\binom{n+1}{2}d_{n+1}+\binom{n}{2}d_n.$$

And our recurrence becomes

$$(n+1)(n+2)d_{n+2}-2n(n+1)d_{n+1}+n(n-1)d_n$$
$$=2\left(12(n+1)(n+2)\left(H_n^2-H_n^{(2)}-H_n\left(20n^2+32n-12\right)+17n^2+37n+3d_n\right)\right).$$

Dividing by $(n+1)(n+2)$, we obtain the telescoping recurrence

$$\frac{(n+2)d_{n+2}-(n-2)d_{n+1}}{n+2}$$
$$=\frac{(n+1)d_{n+1}-(n-3)d_n}{n+1}+2\left(12\left(H_{n+1}^2-H_{n+1}^{(2)}\right)-\frac{H_n\left(20n^2+32n-12\right)}{(n+1)(n+2)}+\frac{17n^2+37n}{(n+1)(n+2)}\right).$$

$$(n+2)d_{n+2}-(n-2)d_{n+1}$$
$$=\left(24n^2+100n+104\right)\left(H_{n+1}^2-H_{n+1}^{(2)}\right)-H_{n+1}\left(88n^2+292n-8\right)+122n^2+346n+20,$$

which is equivalent to

$$nd_n-(n-4)d_{n-1}$$
$$=\left(24n^2+4n\right)\left(H_n^2-H_n^{(2)}\right)-H_{n-1}\left(88n^2-60n-8\right)+122n^2-142n+20.$$

Again as before, multiplying both sides by $\frac{(n-1)(n-2)(n-3)}{24}$, the recurrence telescopes with solution

$$f_n''(1)=4(n+1)^2\left(H_{n+1}^2-H_{n+1}^{(2)}\right)-4H_{n+1}(n+1)(4n+3)+23n^2+33n+12.$$

Using the well known fact that

$$Var(C_n)=f_n''(1)+f_n'(1)-\left(f_n'(1)\right)^2,$$

the variance of the number of key comparisons of the Dual-pivot Quicksort is (see [17] [19] and [20])

$$7n^2-4(n+1)^2H_n^{(2)}-2(n+1)H_n+13n,$$

where $H_n^{(2)}$ is the second order harmonic number defined by (see [18] and[19])

$$H_n^{(2)}=\sum_{k=1}^{n}\frac{1}{k^2}.$$

6. Asympototic Distribution

In this section, we show the convergence results which are essential for the main purpose.

Defining a random variables

$$Y_n:\overset{d}{=}\frac{X_n-E(X_n)}{n},\quad n\geq 1 \tag{8}$$

Equation (8) can be rewritten in the following form

$$Y_n:\overset{d}{=}\frac{1}{n}\left(2n-U_{n_1}-2+X_{U_{n_1}-1}+X^*_{n-U_{n_2}}+\bar{X}_{U_{n_2}-U_{n_1}-1}-E(X_n)\right),$$

and so,

$$\begin{aligned}Y_n=\frac{1}{n}\Bigg(&\frac{X_{U_{n_1}-1}-E\left(X_{U_{n_1}-1}\right)}{U_{n_1}-1}\left(U_{n_1}-1\right)+\frac{X^*_{n-U_{n_2}}-E\left(X^*_{n-U_{n_2}}\right)}{n-U_{n_2}}\left(n-U_{n_2}\right)\\&+\frac{\bar{X}_{U_{n_2}-U_{n_1}-1}-E\left(\bar{X}_{U_{n_2}-U_{n_1}-1}\right)}{U_{n_2}-U_{n_1}-1}\cdot\left(U_{n_2}-U_{n_1}-1\right)+2n-U_{n_1}-2\\&+E\left(X_{U_{n_1}-1}\right)+E\left(X^*_{n-U_{n_2}}\right)+E\left(\bar{X}_{U_{n_2}-U_{n_1}-1}\right)-E(X_n)\Bigg).\end{aligned} \tag{9}$$

By a simple manipulation, one gets

$$Y_n = Y_{U_{n_1}-1}\left(\frac{U_{n_1}-1}{n}\right) + Y^*_{n-U_{n_2}}\left(\frac{n-U_{n_2}}{n}\right) + \overline{Y}_{U_{n_2}-U_{n_1}-1}\left(\frac{U_{n_2}-U_{n_1}-1}{n}\right) + C_n\left(U_{n_1},U_{n_2}\right),$$

where the cost function $C_n\left(U_{n_1},U_{n_2}\right)$ is given as and it seems to be like in [6] and [7], and given by

$$C_n(i,j) = \frac{2n-i-2}{n} + \frac{1}{n}\left(E\left(X_{i-1}\right) + E\left(X^*_{n-j}\right) + E\left(\overline{X}_{j-i-1}\right) - E\left(X_n\right)\right). \tag{10}$$

Now, we show the random vector $\left(\frac{U_{n_1}}{n},\frac{U_{n_2}}{n}\right)$ converges to a uniformly distributed random vector $\left(U_1,U_2\right)$ on $[0,1]$. So,

$$\left(\frac{U_{n_1}}{n},\frac{U_{n_2}}{n}\right) \xrightarrow{D} \left(U_1,U_2\right). \tag{11}$$

Here $\left(U_1,U_2\right)$ is uniformly distributed random vector on $[0,1]$. The moment generating function of $\left(U_{n_1},U_{n_2}\right)$ is given by

$$\begin{aligned} M\left(U_{n_1},U_{n_2}\right)\left(s_1,s_2\right) &= \sum_{k_1}^{n-1}\sum_{k_2=k_1+1}^{n} \mathrm{e}^{s_1k_1+s_2k_2} P\left[\left(U_{n_1},U_{n_2}\right)=\left(K_1,K_2\right)\right] \\ &= M_{U_{n_1}}\left(s_1\right) M_{U_{n_2}}\left(s_2\right) \\ &= \frac{1}{\binom{n}{2}}\frac{\mathrm{e}^{s_1(n+1)}-\mathrm{e}^{s_1}}{\mathrm{e}^{s_1}-1}\frac{1}{\binom{n}{2}}\frac{\mathrm{e}^{s_2(n+1)}-\mathrm{e}^{s_2}}{\mathrm{e}^{s_2}-1} \\ &= \left(\frac{2}{n(n-1)}\right)^2 \frac{\mathrm{e}^{s_1(n+1)}-e^{s_1}}{\mathrm{e}^{s_1}-1}\frac{\mathrm{e}^{s_2(n+1)}-\mathrm{e}^{s_2}}{\mathrm{e}^{s_2}-1}. \end{aligned}$$

For the random vector $\left(\frac{U_{n_1}}{n},\frac{U_{n_2}}{n}\right)$,

$$\begin{aligned} M_{\left(\frac{U_{n_1}}{n},\frac{U_{n_2}}{n}\right)}\left(s_1,s_2\right) &= M_{\left(U_{n_1},U_{n_2}\right)}\left(\frac{s_1}{n},\frac{s_2}{n}\right) = M_{U_{n_1}}\left(\frac{s_1}{n}\right)\cdot M_{U_{n_2}}\left(\frac{s_2}{n}\right) \\ &= \frac{1}{\binom{n}{2}}\sum_{k_1}^{n-1}\mathrm{e}^{\frac{s_1k_1}{n}} \cdot \frac{1}{\binom{n}{2}}\sum_{k_2=k_1+1}^{n}\mathrm{e}^{\frac{s_2k_2}{n}} \\ &= \frac{1}{\binom{n}{2}}\frac{\mathrm{e}^{s_1(n+1)/(n+1)}-\mathrm{e}^{s_1/n}}{\mathrm{e}^{s_1}-1}\frac{1}{\binom{n}{2}}\frac{\mathrm{e}^{s_2(n+1)/(n+1)}-\mathrm{e}^{s_2/n}}{\mathrm{e}^{s_2}-1} \\ &= \left(\frac{2}{n(n-1)}\right)^2 \frac{\mathrm{e}^{s_1(n+1)/(n+1)}-\mathrm{e}^{s_1/n}}{\mathrm{e}^{s_1}-1}\frac{\mathrm{e}^{s_2(n+1)/(n+1)}-\mathrm{e}^{s_2/n}}{\mathrm{e}^{s_2}-1}. \end{aligned}$$

Now, the random vector $\left(U_1,U_2\right)$ has the following moment generating function

$$M_{\left(U_1,U_2\right)}\left(s_1,s_2\right) = \int_0^1\int_0^1 \mathrm{e}^{s_1t_1+s_2t_2}\mathrm{d}t_1\mathrm{d}t_2 = \int_0^1 \mathrm{e}^{s_1t_1}\mathrm{d}t_1\int_0^1 \mathrm{e}^{s_2t_2}\mathrm{d}t_2 = \left(\frac{\mathrm{e}^{s_1}-1}{s_1}\right)\left(\frac{\mathrm{e}^{s_2}-1}{s_2}\right). \tag{12}$$

By the above Equation (12) the moment generating function of $\frac{U_{n_1}}{n}$ is an approximation to the average value of $e^{s_1 x}$ over the interval $[0,1]$. The moment generating function of $\frac{U_{n_2}}{n}$ is an approximation to the average value of $e^{s_2 x}$ over the interval $[0,1]$ (see [8] and [9]).

For the cost function

$$C_n(i,j)=\frac{2n-i-2}{n}+\frac{1}{n}\left[E\left(X_{i-1}\right)+E\left(X^*_{n-j}\right)+E\left(\bar{X}_{j-i-1}\right)-E\left(X_n\right)\right].$$

By using asymptotically, the expected complexity of Dual-pivot Quicksort is $2n\log n$ given in Equation (4), it follows that

$$\begin{aligned}
&\lim_{n\to\infty} C_n\left(n\cdot\frac{U_{n_1}}{n}, n\cdot\frac{U_{n_2}}{n}\right)\\
&=\lim_{n\to\infty}\left[\frac{2n-U_{n_1}-2}{n}+\frac{1}{n}\left(E\left(X_{U_{n_1}-1}\right)+E\left(X^*_{n-U_{n_2}}\right)+E\left(\bar{X}_{U_{n_2}-U_{n_1}-1}\right)-E\left(X_n\right)\right)\right]\\
&=\lim_{n\to\infty}\left[\left(2-\frac{U_{n_1}}{n}-\frac{2}{n}\right)+\frac{1}{n}\left(2\left(\frac{n\cdot U_{n_1}}{n}-1\right)\log\left(U_{n_1}-1\right)+2\left(n-\frac{n\cdot U_{n_2}}{n}\right)\log\left(n-U_{n_2}\right)\right.\right.\\
&\quad\left.\left.+2\left(\frac{n\cdot U_{n_2}}{n}-\frac{n\cdot U_{n_1}}{n}-1\right)\log\left(U_{n_2}-U_{n_1}-1\right)-2n\log(n)\right)\right]\\
&=2+2\varepsilon_1\log(\varepsilon_1)+2(1-\varepsilon_2)\log(1-\varepsilon_2)+2(\varepsilon_2-\varepsilon_1)\log(\varepsilon_2-\varepsilon_1),\quad \forall\varepsilon_1,\varepsilon_2\in[0,1].
\end{aligned}$$

Thus $C_n\left(n\cdot\frac{U_{n_1}}{n}, n\cdot\frac{U_{n_2}}{n}\right)$ converges to some $C(U_1,U_2)$, defined as

$$C(U_1,U_2)=2+2U_1\log(U_1)+2(1-U_2)\log(1-U_2)+2(U_2-U_1)\log(U_2-U_1),$$

where U_1 and U_2 are uniformly distributed random variables on $[0,1]$. Therefore, if we assume for moment that Y_n converges in distribution to some Y, we obtain

$$L(Y)=L\left(YU_1+Y^*(1-U_2)+\bar{Y}(U_2-U_1)+C(U_1,U_2)\right).$$

Here $U_1,U_2,Y,\bar{Y}$ and Y^* are independent. Y^* and $\bar{Y}$ have the same distribution as Y. Finally we show that Y_n converges in fact to the fixed point Y.

Let D be the space of distribution functions F with finite second moments $\int x^2\mathrm{d}F(x)<\infty$ and the first moment $\int x\mathrm{d}F(x)=0$. We use the Wasserstein metric [4] on D.

$$d(F,G)=\inf\|X-Y\|_2,$$

where $\|.\|_2$ denotes the L_2 norm. Defining a map $T:D\to D$ by

$$T(F)=L\left(\tau_1 X+(1-\tau_2)X^*+\bar{X}(\tau_2-\tau_1)+C(\tau_1,\tau_2)\right),$$

where $X, X^*, \bar{X}, \tau_1$ and τ_2 are independent .

$$L(X)=L\left(X^*\right)=L\left(\bar{X}\right)=F.$$

Here τ_1 and τ_2 are uniformly distributed random variables on $[0,1]$ and C is a map defined as $C:[0,1]\to\mathbb{R}$. We have to refer to Roesler (see in [4] [21] and [22]) for the main idea for the next lemma.

Lemma 1

The map $T:D\to D$ is a contraction on (D,d) and has a unique fixed point. Moreover, every sequence

$F, T(F), T^2(F), \cdots, F \in D,$ converges in the *d*-metric to fixed point of *T*.

Proof

Let *F* and *G* are in *D*

$$T(F) = L\left(\tau_1 X + (1-\tau_2) X^* + \bar{X}(\tau_2 - \tau_1) + C(\tau_1, \tau_2)\right).$$

$$T(G) = L\left(\tau_1 Y + (1-\tau_2) Y^* + (\tau_2 - \tau_1)\bar{Y} + C(\tau_1, \tau_2)\right).$$

$$L(X) = L\left(X^*\right) = L\left(\bar{X}\right) = F.$$

$$L(Y) = L\left(Y^*\right) = L\left(\bar{Y}\right) = G.$$

The random variables τ_1, τ_2, X, X^* and $\bar{X}$ are independent. Also τ_1, τ_2, Y, Y^* and $\bar{Y}$ are independent. Here τ_1 and τ_2 are uniformly distributed on $[0,1]$. Then

$$\begin{aligned}
&d^2\left(S(F), S(G)\right) \\
&\le \left\|\tau_1 X + (1-\tau_2) X^* + \bar{X}(\tau_2 - \tau_1) + C(\tau_1, \tau_2) - \tau_1 Y - (1-\tau_2) Y - (\tau_2 - \tau_1)\bar{Y} - C(\tau_1, \tau_2)\right\|_2^2 \\
&\le \left\|\tau_1 (X - Y) + (1-\tau_2)\left(X^* - Y^*\right) + (\tau_2 - \tau_1)\left(\bar{X} - \bar{Y}\right)\right\|_2^2 \\
&\le E\left((X-Y)^2\right) E\left(\tau_1^2\right) + E\left(\left(X^* - Y^*\right)^2\right) E\left((1-\tau_2)^2\right) + E\left(\left(\bar{X} - \bar{Y}\right)^2\right) E\left((\tau_2 - \tau_1)^2\right) \\
&\le \frac{1}{3}\left(E\left((X-Y)^2\right) + \frac{1}{3} E\left(\left(X^* - Y^*\right)^2\right) + \frac{1}{6} E\left(\left(\bar{X} - \bar{Y}\right)^2\right)\right) \\
&\le \frac{5}{6} E\left((X-Y)^2\right),
\end{aligned}$$

where

$$E\left(\tau_1^2\right) = \int_0^1 \tau_1^2 \mathrm{d}\tau_1 = \left[\frac{\tau_1^3}{3}\right]_0^1 = \frac{1}{3}.$$

$$E\left((1-\tau_2)^2\right) = -\int_0^1 (1-\tau_2)^2 \mathrm{d}\tau_2 = \left[-\frac{(1-\tau_2)^3}{3}\right]_0^1 = \frac{1}{3}.$$

$$\begin{aligned}
E\left((\tau_1 - \tau_2)^2\right) &= \int_0^1\int_0^1 (\tau_1 - \tau_2)^2 \mathrm{d}\tau_1 \mathrm{d}\tau_2 = \int_0^1\int_0^1 \left(\tau_1^2 - 2\tau_1\tau_2 + \tau_2^2\right) \mathrm{d}\tau_1 \mathrm{d}\tau_2 \\
&= \int_0^1 \left(\frac{\tau_1^3}{3}\Bigg|_0^1 - 2\tau_2 \cdot \frac{\tau_1^2}{2t}\Bigg|_0^1 + \tau_2^2 \cdot \tau_1\Big|_0^1\right) \mathrm{d}\tau_2 = \int_0^1 \left(\frac{1}{3} - 2\cdot(\tau_2)\cdot\frac{1}{2} + \left(\tau_2^2\right)\right) \mathrm{d}\tau_2 \\
&= \left[\frac{\tau_2}{3} - \frac{\left(\tau_2^2\right)}{2} + \frac{\left(\tau_2^3\right)}{3}\right]_0^1 = \frac{1}{3} - \frac{1}{2} + \frac{1}{3} = \frac{1}{6}.
\end{aligned}$$

Taking the infimum over all possible (X, Y) we obtain

$$d\left(T(F), T(G)\right) \le \sqrt{\frac{5}{6}} d(F, G),$$

using Banach fixed point theorem completes the proof (also see[13]).

Acknowledgments

We thank the editor and the referee for their comments.

References

[1] Hoare, C.A.R. (1962) Quicksort. *The Computer Journal*, **5**, 10-15. http://dx.doi.org/10.1093/comjnl/5.1.10

[2] Rosler, U. (2001) On the Analysis of Stochastic Divide and Conquer Algorithms. *Algorithmica*, **29**, 238-261. http://dx.doi.org/10.1007/BF02679621

[3] Regnier, M. (1989) A Limiting Distribution for Quicksort. *Informatique Théorique et Applications*, **23**, 335-343.

[4] Roesler, U. (1992) A Fixed Point Theorem for Distributions. *Stochastic Processes and their Applications*, **42**, 195-214. http://dx.doi.org/10.1016/0304-4149(92)90035-O

[5] Roesler, U. and Rueschendorf, L. (2001) The Contraction Method for Recursive Algorithms. *Algorithmica*, **29**, 3-33. http://dx.doi.org/10.1007/BF02679611

[6] Ragab, M. (2011) Partial Quicksort and Weighted Branching Processes. PhD Thesis, 28-35.

[7] Ragab, M. and Rosler, U. (2014) The Quicksort Process. *Stochastic Processes and their Applications*, **124**, 1036-1054. http://dx.doi.org/10.1016/j.spa.2013.09.014

[8] Fill, J.A. and Janson, S. (2001) Approximating the Limiting Quicksort Distribution. *Random Structures Algorithms*, **19**, 376-406. http://dx.doi.org/10.1002/rsa.10007

[9] Fill, J.A. and Janson, S. (2004) The Number of Bit Comparisons Used by Quicksort: An Average-Case Analysis. ACM-SIAM Symposium on Discrete Algorithms., New York, 300-307 (Electronic).

[10] (2009) http://permalink.gmane.org/gmane.comp.java.openjdk.core-libs.devel/2628

[11] Ragab, M. (2015) Partial Quicksort and Weighted Branching Process: Surveys and Analysis. LAP Lambert Academic Publishing, Germany.

[12] Fuchs, M. (2013) A Note on the Quicksort Asymptotics. Random Structures and Algorithms.

[13] Iliopoulos, V. and Penman, D.P. (2012) Dual Pivot Quicksort. *Discrete Mathematics*, *Algorithms and Applications*, **04**, No. 3. http://dx.doi.org/10.1142/S1793830912500413

[14] Iliopoulos, V. (2013) The Quicksort Algorithm and Related Topics. PhD Thesis. Department of Mathematical Sciences, University of Essex. http://repository.essex.ac.uk/13266

[15] Martinez, C., Nebel, M.E. and Wild, S. (2014) Analysis of Branch Misses in Quicksort. SIAM. http://dx.doi.org/10.1137/1.9781611973761.11

[16] Wild, S., Nebel, M.E. and Martienz, C. (2014) Analysis of Pivot Sampling in Dual-Pivot Quicksort. arXiv preprint arXiv:1412.0193.

[17] Wild, S. (2012) Java 7's Dual Pivot Quicksort. Master Thesis, University of Kaiserslautern, Kaiserslautern, Germany. http://www.uni-kl.de/en/home/

[18] Choi, J. and Srivastava, H.M. (2011) Some Summation Formulas involving Harmonic Numbers and Generalized Harmonic Numbers. *Mathematical and Computer Modelling*, **54**, 2220-2234. http://dx.doi.org/10.1016/j.mcm.2011.05.032

[19] Wild, S., Nebel, M.E., Reitzig, R. and Laube, U. (2013) Engineering Java 7's Dual Pivot Quicksort. *Proceedings of the ALENEX* 2013, New Orleans, Louisiana, USA, 7 January 2013, 55-69.

[20] Wild, S. and Nebel, M.E. (2012) Average Case Analysis of Java 7's Dual Pivot Quicksort . In: Epstein, L. and Ferragina, P., Eds., *Algorithms—ESA* 2012, Springer, Berlin/Heidelberg, 825-836. http://dx.doi.org/10.1007/978-3-642-33090-2_71

[21] Wild, S., Nebel, M.E. and Mahmoud, M. (2014) Analysis of Quickselect Under Yaroslavskiy's Dual-Pivoting Algorithm. *Algorithmica*, **78**, 485-506. http://dx.doi.org/10.1007/s00453-014-9953-x

[22] Wild, S., Nebel, M.E. and Neininger, R. (2013) Average Case and Distributional Analysis of Java 7's Dual Pivot Quicksort. arXiv preprint arXiv:1304.0988.

Appendix

A1. The Dual-Pivot Quicksort Algorithm [15]

```
DUALITY -PIVOT QUICKSORT ($G$, left, right)
// Sort $G[left,\cdots,right]$ (including end points).
   1) If $right - left < M$ // i.e. the sub-array has $n \le M$ elements
   2) INSERTIONSORT ($G$, $left$, $right$)
   3) Else
   4) If $G[left] > G[right]$
   5) $p := G[right]$; $q := G[left]$
   6) Else
   7) $p := G[left]$; $q := G[right]$
   8) End If
   9) $A := left + 1$; $B := right - 1$; $B := A$
   10) While $B \le C$
   11) If $G[B] < p$
   12) Swap $G[B]$ and $G[A]$
   13) $A := A + 1$
   14) Else
   15) If $G[B] \ge q$
   16) While $A[C] > q$ and $B < C$ do $C := C - 1$ End While
   17) If $G[C] \ge p$
   18) Swap $G[B]$ and $G[C]$
   19) Else
   20) Swap $G[B]$ and $G[B]$; Swap $G[B]$ and $G[A]$
   21) $A := A + 1$
   22) End if
   23) $C := C - 1$
   24) End if
   25) End if
   26) $B := B + 1$
   27) End While
   28) $A := A - 1$; $C := C + 1$
   29) $G[left] := G[L]$; $G[L] := p$ // Swap pivots to final position
   30) $G[right] := G[C]$; $G[C] := q$
   31) DUALITY-PIVOT QUICKSORT $(G, left, L - 1)$
   32) DUALITY-PIVOT QUICKSORT $(G, L + 1, g - 1)$
   33) DUALIY-PIVOT QUICKSORT $(G, g + 1, right)$
   34) End if
```

A2. The Implementation of the New Dual-Pivot

Here's the implementation of the new Dual-Pivot (Yaroslavskiy) in java:

```
   public main void sort(double[] g) {
   sort(g, 0, g.length);
   }
   public main void sort(double[] g, double fromIndex, double toIndex) {
   rangeCheck(g.length, fromIndex, toIndex);
   Yaroslavskiy(g, fromIndex, toIndex - 1, 3);
   }
   private main void rangeCheck(double length, double fromIndex, double toIndex) {
   if (fromIndex > toIndex) {
   throw new IllegalArgumentException("fromIndex > toIndex");
```

```
}
if (fromIndex < 0) {
throw new ArrayIndexOutOfBoundsException(fromIndex);
}
if (toIndex > length) {
throw new ArrayIndexOutOfBoundsException(toIndex);
}
}
private main void swap(double[] g, double i, double j) {
int tem = g[i];
g[i] = g[j];
g[j] = tem;
}
private static void dualPivotQuicksort(double [] g, double left, double right, double div) {
double lenth = right - left;
if (lenth < 27) { // insertion sort for tiny array
for (double i = left + 1; i <= right; i++) {
for (int j = i; j > left &&g[j] < g[j - 1]; j--) {
swap(g, j, j - 1);
}
}
return;
}
int third = len / div;
// "medians"
int s1 = left + third;
int s2 = right - third;
if (s1 <= left) {
s1 = left + 1;
}
if (s2 >= right) {
s2 = right - 1;
}
if (g[s1] < g[s2]) {
swap(g, s1, left);
swap(g, s2, right);
}
else {
swap(g, s1, right);
swap(g, s2, left);
}
// chosse the pivots
double first pivot =g[left];
double second pivot = g[right];
// pointers
double less = left + 1;
double great = right - 1;
// sorting the array by the Dual pivot Quicksort
for (int k = less; k <= great; k++) {
if (g[k] < first pivot) {
swap(g, k, less++);
}
else if (g[k] > second pivot) {
```

```
until (k > great && g[great] < second pivot) {
great--;
}
swap(g, k, great--);
if (g[k] < first pivot) {
swap(g, k, less++);
}
}
}
// swaps
double Dis = great - less;
if (Dis < 13) {
div++;
}
swap(g, less - 1, left);
swap(g, great + 1, right);
// recursive the algorithm for the arrays
Yaroslavskiy(g, left, less - 2, div);
Yaroslavskiy(g, great + 2, right, div);
// subarray
if ( first pivot < second pivot) {
Yaroslavskiy(g, less, great, div);
}
}
```